LAD 23

Lustre Admin & Dev Workshop

Bordeaux France 04/10 - 06/10/2023

Whamcloud

Empowering Lustre Performance Evolution through IO500

Shuichi Ihara

ddn

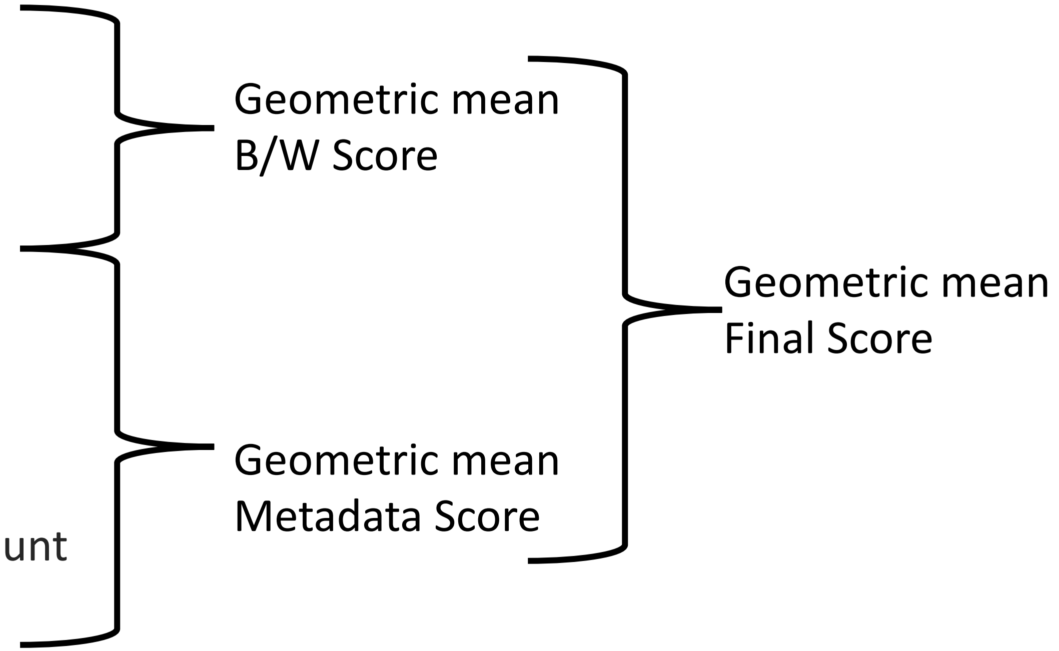
IO Benchmark and IO500

- Understanding IO performance of storage systems
  - IO Benchmark is one of critical storage requirements
  - Standardized tests and performance comparisons are also important
    - e.g. old systems vs new systems, different hardware configurations, etc
- IO500 - https://io500.org/
  - A standard IO Benchmark suite for HPC
    - Easy to use, assesses IO bandwidth, metadata ops, and search performance
    - Includes "Easy" (Hero) and "Hard" (Anti-Hero) tests
    - Comprehensive evaluation, no single performance criterion
  - Publication
    - Results shared in ranking lists and started at SC17 in Nov 2017
    - Updates biannually, like other xxx500 lists
    - Categories: "Production" "Research" "Full” and "10 Client"
    - Accepts submissions from various environments (Production, Test system, on-premise, Cloud, etc.)

IO500

Benchmark components

- IOR (Write and Read)
  - ○ Easy : FPP (File per process), various IO size allowed
  - ○ Hard : Shared file, Interleaved, Fixed IO size (47,008 bytes)
- mdtest (write, stat, read* and delete)
  - ○ Easy : Individual directories, zero byte files
  - ○ Hard : Shared Directory, small files (3,901 bytes)
- Find
  - ○ Search 3,901-byte files from all created files and print total count
  - ○ External tools allowed



Geometric mean
B/W Score

Geometric mean
Metadata Score

Geometric mean
Final Score

Other Rules

- Writes must be longer than 5 minutes and stored on persistent devices
- Avoid client caches (e.g. Stride MPI ranks in next operations. "`-C -Q 1`” for IOR)

*read in `mdtest` runs only for `mdtest-hard`

Leveraging IO500

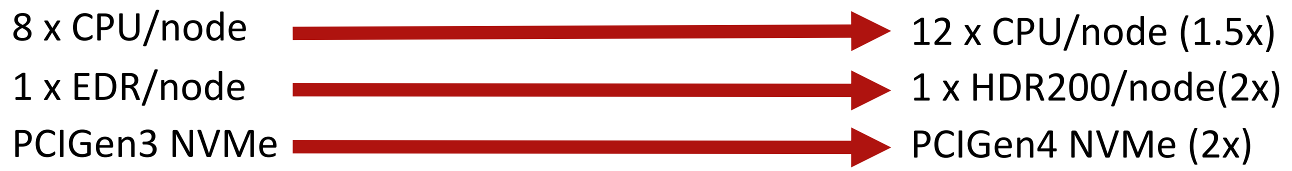
- Consistent and historical benchmark
  - Performance regression tests
  - Identifying performance challenges and demonstrating performance improvements
- Enhancing performance efficiency
  - Achieving high I/O performance with small hardware resources
    - Maximize I/O performance by 10 clients
    - Not only HPC, but also for AI/ML - A large GPU node (e.g. 8 x GPU, 2 x 400Gbps network)
- Exploring I/O Performance through IO500
  - Identifying bottlenecks in both Lustre and I/O subsystem overall
  - Adapting to real performance challenges in the production system
  - Improve I/O traceability
  - Refer useful `mdtest` and `ior` command for your storage requirements
    - “`mdtest -u`” (cached) and “`mdtest -u -N 1`” (non-cached) in IO500 are totally different workload

Our Sustained performance enhancements

Whamcloud

Storage Platform ES400NV ES400NVX ES400NVX2

10 x Client
1 x CPU, 96GB RAM
1 x HDR100



8 x CPU/node → 12 x CPU/node (1.5x)

1 x EDR/node → 1 x HDR200/node(2x)

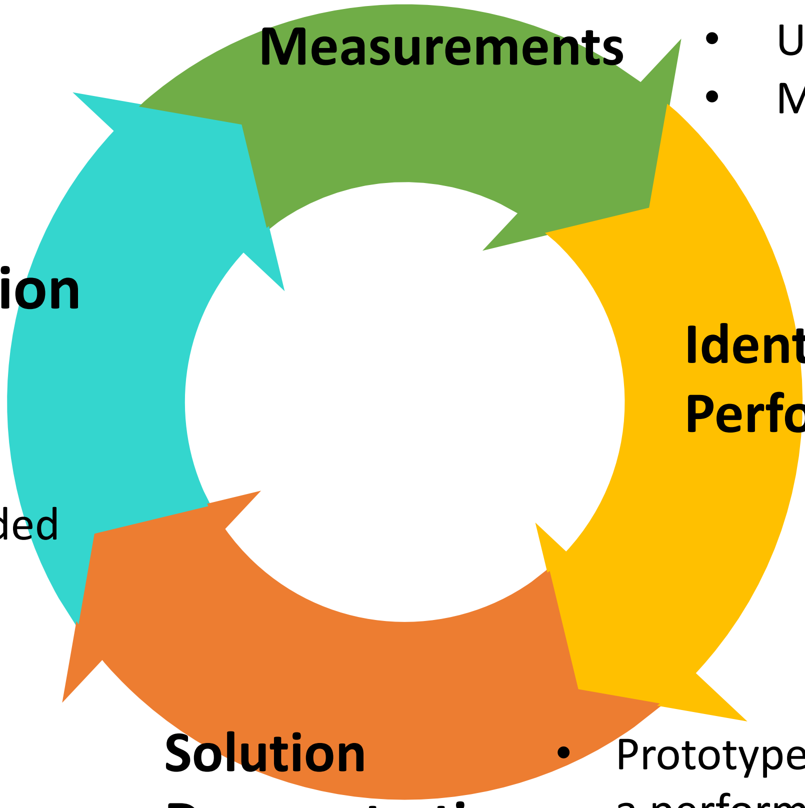
PCIGen3 NVMe → PCIGen4 NVMe (2x)

Performance improvements go beyond what hardware upgrades can achieve

	Pre-SC19	SC19	ISC20	ISC22	SC22	ISC23	ISC23/PreSC19
IOR Easy Write	25.88	28.62	37.56	55.95	58.07	57.88	2.2x
IOR Easy Read	39.94	41.72	45.95	83.86	77.56	79.08	2.0x
IOR Hard Write	2.78	2.96	2.77	5.02	5.27	5.38	2.0x
IOR Hard Read	8.99	42.19	40.81	39.73	49.36	50.77	5.6x
Find	1,735.41	810	1,698.00	6,248.55	12628.78	13,229.11	7.6x
Mdtest Easy Write	143.88	152.84	157.22	270.04	312.9	344.70	2.3x
Mdtest Easy Stat	455.03	451.97	453.51	740.01	1,278.50	1,276.31	2.8x
Mdtest Easy Delete	88.52	132.76	135.09	223.61	272.64	311.16	3.5x
Mdtest Hard Write	32.33	79.65	90.47	119.41	157.4	199.36	6.1x
Mdtest hard Read	44.92	172.59	169	194.33	238.82	391.09	8.7x
Mdtest Hard Stat	20.41	449.93	446.75	514.36	1,214.03	1,105.33	54.1x
Mdtest Hard Delete	16.35	75.15	76.94	101.98	122.44	112.58	6.8x
Bandwdith	12.68	19.65	21.02	31.10	32.90	33.43	2.6x
IOPS	91.41	207.62	232.69	368.48	544.23	603.39	6.6x
Score	34.05	63.87	69.93	107.05	133.81	142.03	4.1x

https://io500.org/submissions/view/657

Successful Lustre Performance Improvement cycle



Performance Measurements

- Run IO500
- Understanding system and Lustre behaviors
- Minimized reproducer

Identifying Performance Bottlenecks

- Lustre limitations and others
- Bottleneck was hidden, but now exposed
- Certain areas were relatively unexplored

Solution Demonstration

- Prototype proposal to address a performance challenge
- Re-run IO500 on 10 nodes
- New performance demonstration

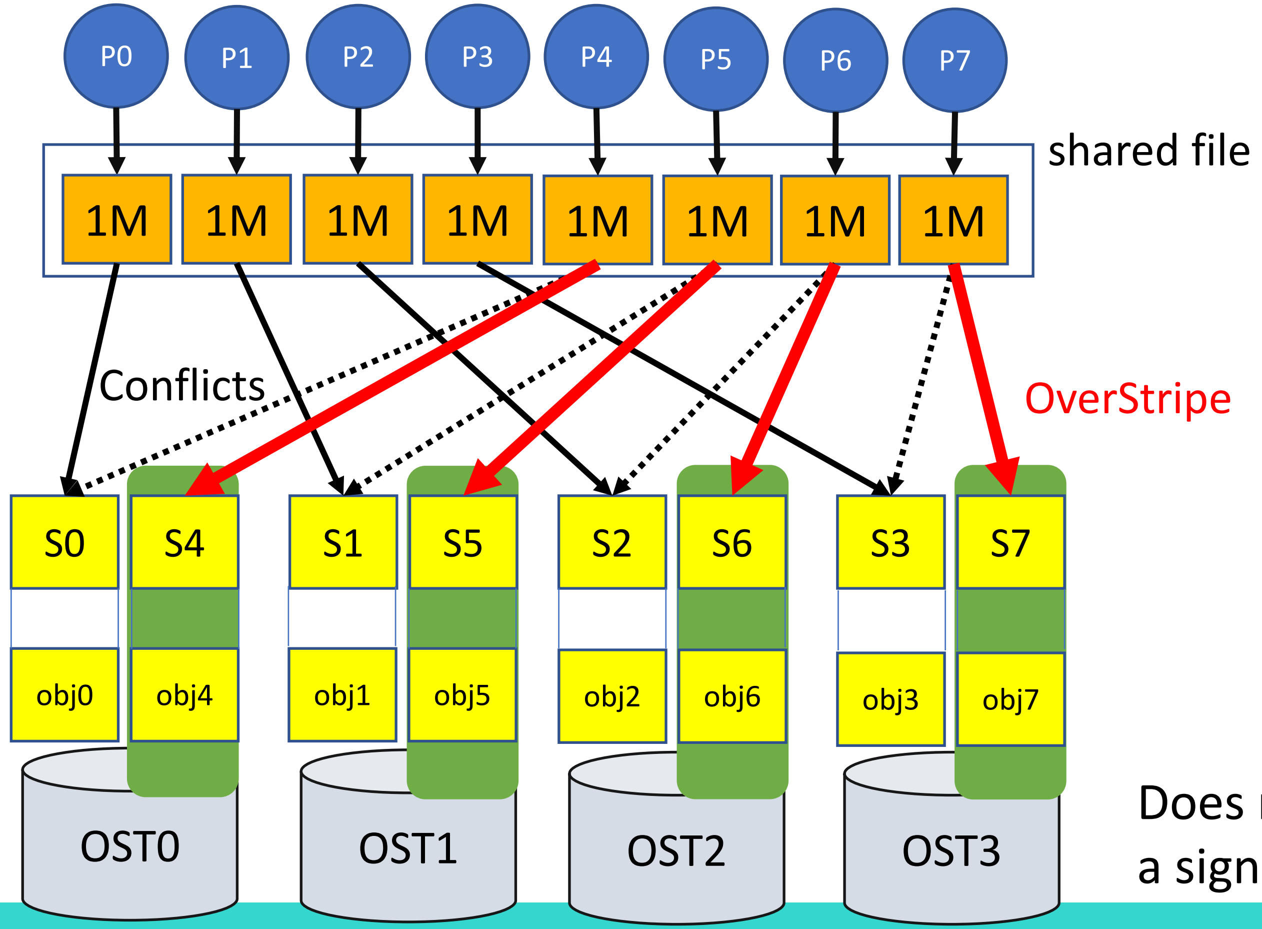
Productization

- Full regression tests
- Review and merge patches
- Enable at large production systems and tunning if needed

Lustre OverStripe (Lustre-2.13)

Whamcloud

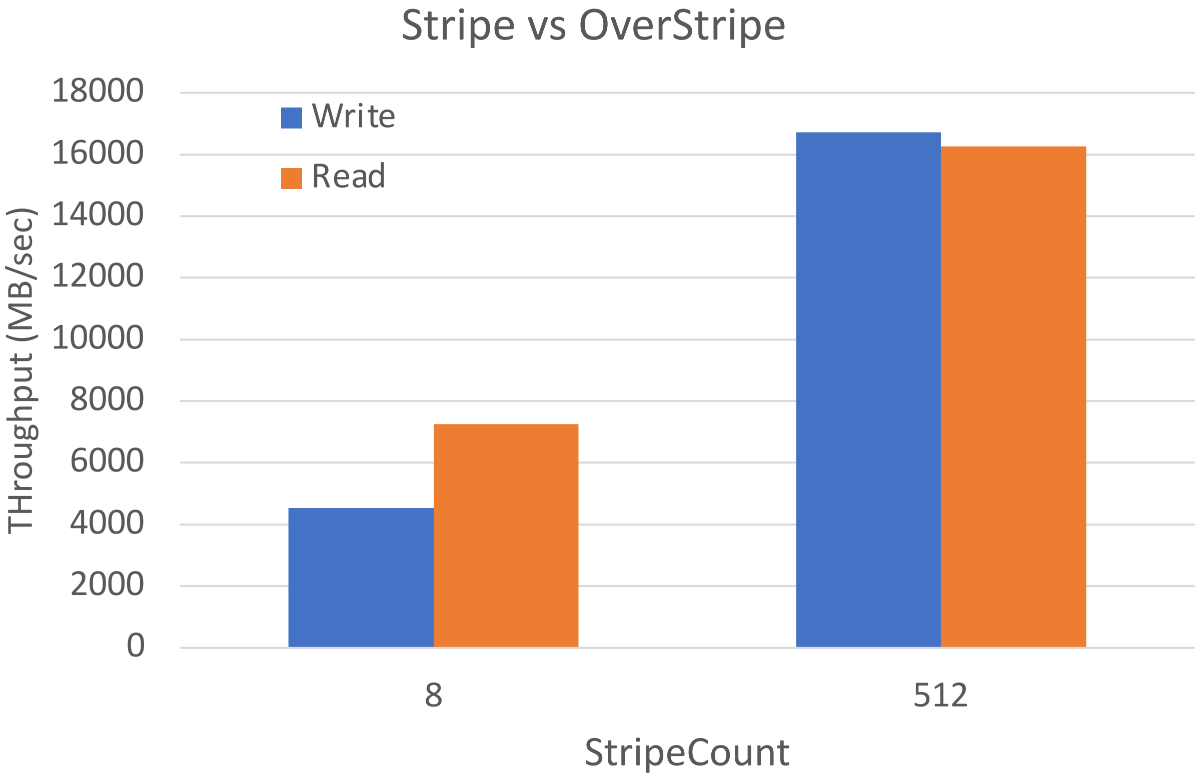
```
# lfs setstripe -c 4 /lustre/file (Lustre Regular Stripe)
# lfs setstripe -C 8 /lustre/file (OverStripe)
```



1MB single shared file

```
# ior -w -r -C -g -i 3 -vv -s 13000 -b 1m -t 1m -a POSIX –e
```

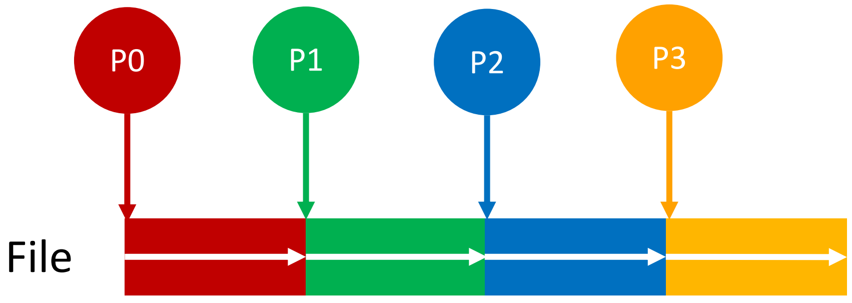
ES7990(160 x HDD, 2 x OSS, 8 x OST), 32 clients



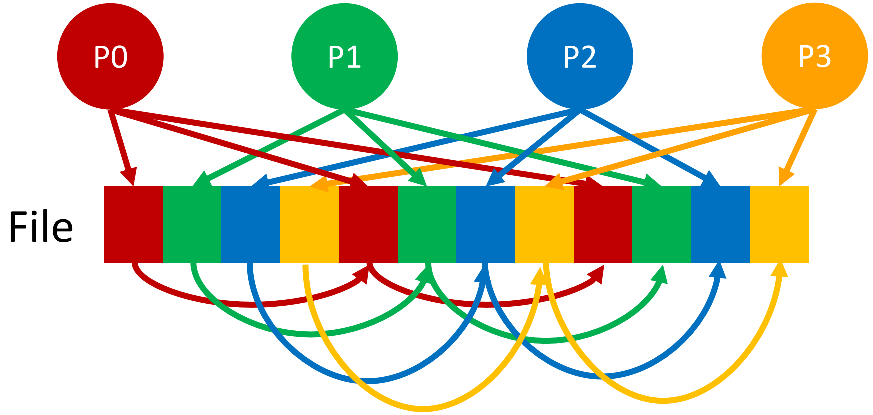
Does not solve `ior-hard-write` completely, but offers a significant performance improvement for a single shared file

Improvements of Lustre ReadAhead (Lustre-2.14)

- Accurate detection of I/O patterns
  - Readahead has been working well for sequential reads.
  - Support new IO pattern ”Stride Read” for a single shared file
- Changed page based index to bytes offset
  - Support unaligned page (e.g. 47008 byte in `ior-hard-read`)
  - Avoid many small page RPCs and readahead windows reset
    - Improve readahead cache hit rate

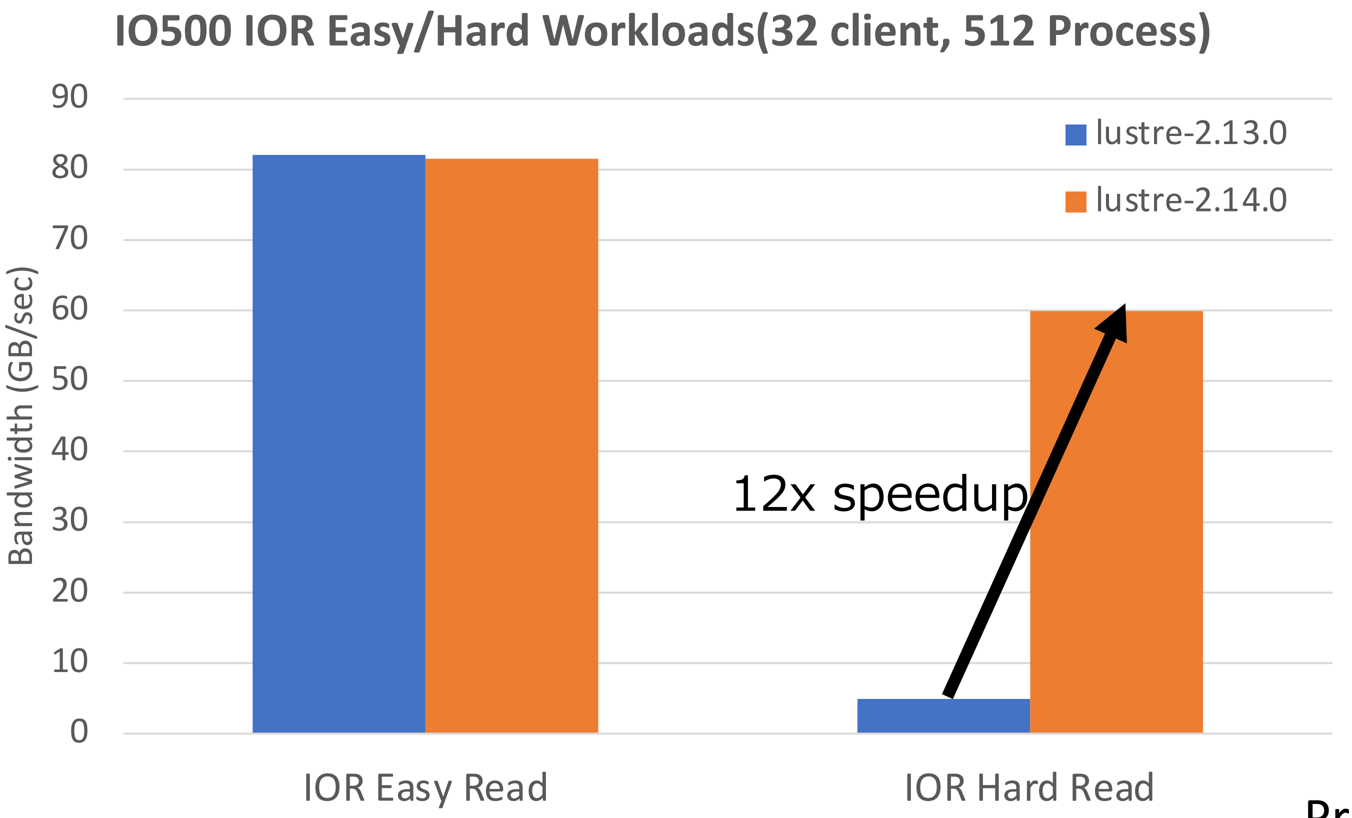


Sequential Read



Stride Read

Performance comparisons of Lustre-2.13 and lustre-2.14



Readahead stats for `ior-hard-read`

Lustre-2.13

```
# lctl get_param llite.*.read_ahead_stats
llite.exafs-ffff9b96c1349800.read_ahead_stats=
hits                          3340631 samples [pages]
misses                        32901120 samples [pages]
```

Readahed Cache Hit rate: 9%

Lustre-2.14

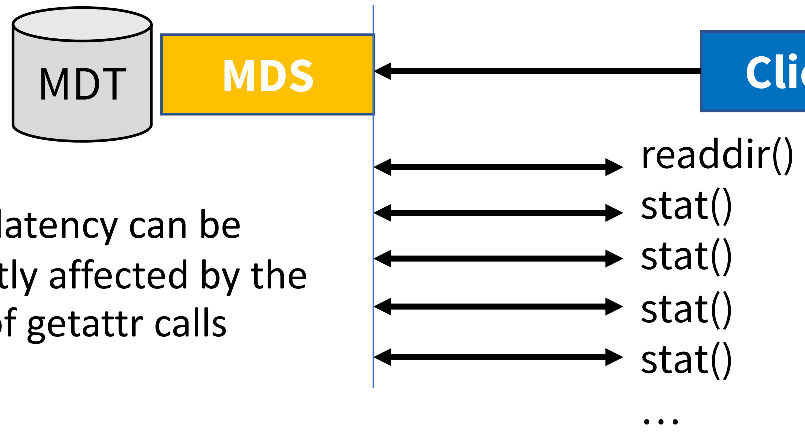
```
llite.exafs-ffff9b96b8117000.read_ahead_stats=
hits                          33616605 samples [pages]
misses                        4444696 samples [pages]
```

Readahed Cache Hit rate: 88%

Prefetched data by readahead hits expected next read

Lustre Batched RPCs for Statahead (Lustre-2.16)

Traditional statahead



MDT

MDS

Client

readdir()

stat()

stat()

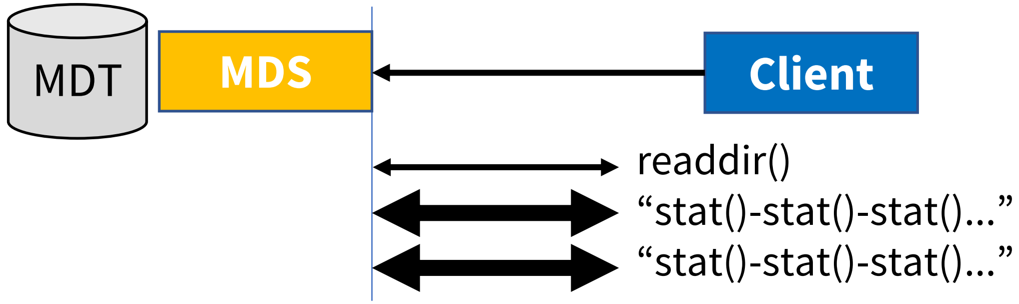
stat()

stat()

…

Network latency can be significantly affected by the number of getattr calls

Batched statahead



MDT

MDS

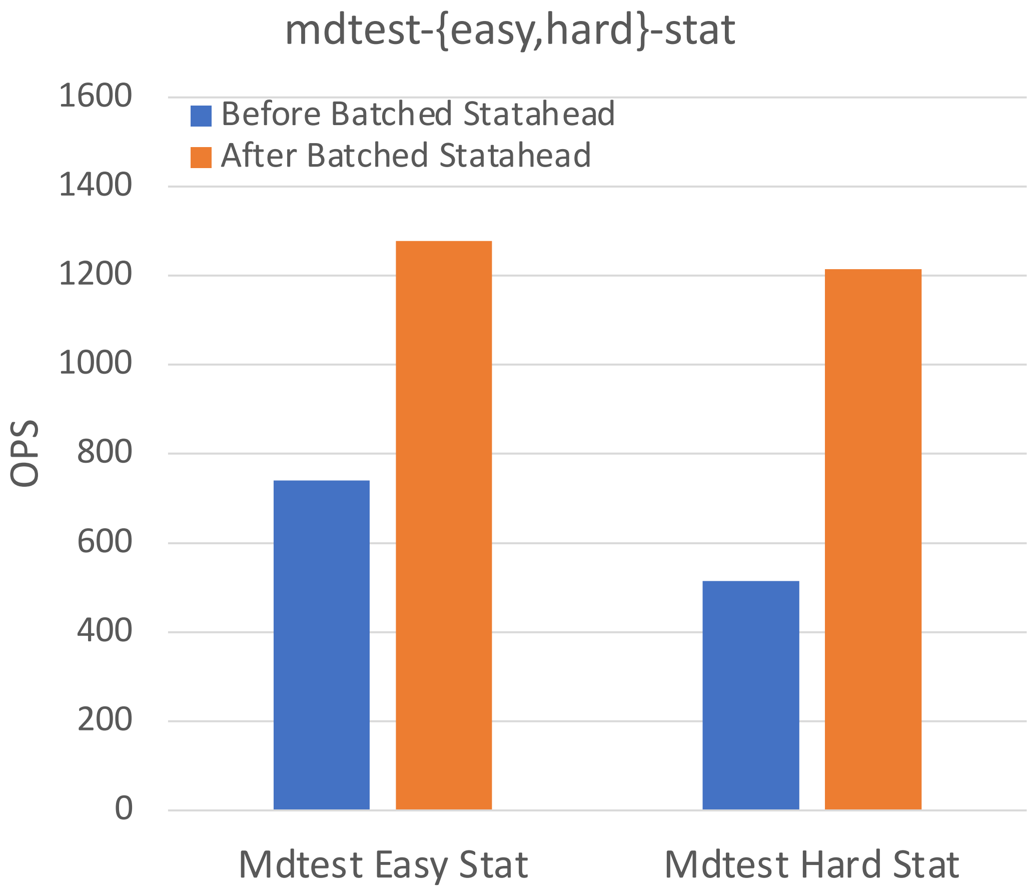
Client

readdir()

“stat()-stat()-stat()…”

“stat()-stat()-stat()…”

Aggregate multiple getattr RPCs and send them as a batched large request to severs



Additional Lustre Performance Enhancements

- Automated MDT usage/space balancing (Lustre-2.14/Lustre-2.15)
  - Each unique sub directory can be automatically assigned to an MDT and avoiding striped directory
- Metadata OverStriping (LU-12273)
  - Similar concept to OST OverStripe, but it allows MDT stripe counts > MDTs

Other Tips of Performance Improvements for IO500

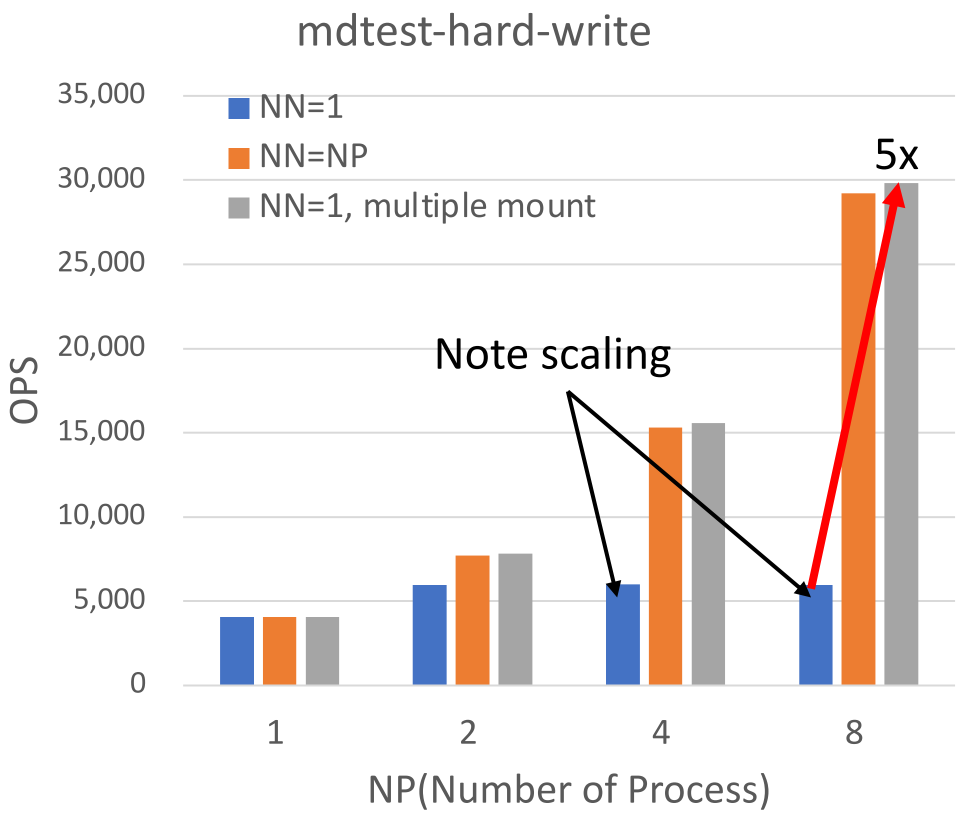
- Developed an external tool for metadata scan/search
  - Alternative tool to "`lfs find`", "`find`" and "`pfind`" that allows for scanning MDT directly without relying on Lustre clients
  - 7x performance improvements compared to "`pfind`"
- Linux kernel for Lustre server
  - Upgrading from RHEL7.x to RHEL8.x servers improved metadata performance by 25-30%
- Linux kernel for Lustre client
  - VFS Parallel Lookup (Supported since kenrel-4.7) speeds up `stat()` operation for a shared directory (`mdtest-hard-stat`)
  - There are still performance limitations with parallel modifications to a shared directory through VFS
    - Neil Brown submitted a proposal to the upstream kernel "VFS: Support Parallel Updates in a Single Directory"
    - Using multiple mountpoints in containers from Lustre client is a workaround
      - Commonly used in HPC/Cloud today to run multiple jobs on a single compute/GPU node

Multiple mount points on Lustre client

- Lustre allows multiple mount points against a single Lustre filesystem
  - Many use cases are exists (security, sub directory mounts, high performance GPU client)
  - Some Lustre parameters need to be adjusted
- Enables multiple mount points for IO500
  - Mount Lustre on different mount points
    - ○ e.g. /mnt/lustre_0, /mnt/lustre_1, /mnt/lustre_2, …
  - Configure singularity with multiple mount points for MPI

```
io500_mpirun="mpirun”
io500_mpiargs="singularity.sh -B /usr/mpi -B /usr/lib64
-B /sys/class/infiniband_verbs -B /bin -B /sbin
-B /etc centos8.sif”

#/bin/sh
# singularity.sh
MNT_ID=$((OMPI_COMM_WORLD_RANK % 8))
singularity exec --bind /mnt/lustre_${MNT_ID}:/mnt/lustre $*
```



5x

Note scaling

Summary

- Lustre performance has been proven on large production HPC systems at numerous sites
  - IO500 is an example benchmark metric, but it's not the only one
  - In addition to performance, high RAS capability are necessary in large-scale HPC systems
  - On the other hand, IO500 opened an door for new Lustre performance evolutions in HPC/AI and more
- What’s next?
  - Multiple efforts are underway to investigate for unaligned IO (`ior-hard-write`) performance improvements
    - DIO support for unaligned IO
    - Enabling delayed allocation in osd-ldiskfs
  - Cross-file Readahead
    - Expect `mdtest-hard-read` performance boosts
    - It also helps many small file read workload
  - Consider upgrading the Linux kernel for servers (e.g. RHEL9)

Stay tuned!

Whamcloud